BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

As rescanning documents *will not* correct images, please do not report the images to the Image Problem Mailbox.

PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant:	Myun-Hoon SUNWOO	Examiner: Not Yet Assigned
Serial No.:	10/608,511	Group Art Unit: 2631
Filed:	June 27, 2003	Docket: 678-1202
For:	MODULATION APPARATUS USING MIXED-RADIX FAST FOURIER TRANSFORM	Dated: March 25, 2004

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

TRANSMITTAL OF PRIORITY DOCUMENT

Sir:

Enclosed is a certified copy of Korean Appln. No. 2003-0042357 filed on June 27, 2003, from which priority is claimed under 35 U.S.C. §119.

Respectfully submitted,

Paul J. Farrell
Registration No. 33,494
Attorney for Applicant

DILWORTH & BARRESE, LLP
333 Earle Ovington Boulevard
Uniondale, New York 11553
(516) 228-8484

CERTIFICATE OF MAILING UNDER 37 C.F.R. § 1.8 (a)

I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail, postpaid in an envelope, addressed to the: Commissioner of Patents, P.O. Box 1450, Alexandria, VA 22313-1450 on March 25, 2004.

Dated: March 25, 2004

Paul J. Farrell

대 한 민 국 특 허 청

KOREAN INTELLECTUAL PROPERTY OFFICE

별첨 사본은 아래 출원의 원본과 동일함을 증명함.

This is to certify that the following application annexed hereto is a true copy from the records of the Korean Intellectual Property Office.

출 원 번 호 : Application Number	10-2003-0042357
출 원 년 월 일 : Date of Application	2003년 06월 27일 JUN 27, 2003
출 원 인 : Applicant(s)	삼성전자주식회사 외 1명 SAMSUNG ELECTRONICS CO., LTD., et al.

2003 년 08 월 01 일

특 허 청

COMMISSIONER

【서지사항】

【서류명】	특허출원서
【권리구분】	특허
【수신처】	특허청장
【참조번호】	0001
【제출일자】	2003.06.27
【국제특허분류】	H04L
【국제특허분류】	H03M
【발명의 명칭】	고속 퓨리에 변환을 이용한 혼합-기수 방식의 변조 장치
【발명의 영문명칭】	MODULATING APPARATUS FOR USING FAST FOURIER TRANSFORM OF MIXED-RADIX SCHEME
【출원인】	
【명칭】	삼성전자 주식회사
【출원인코드】	1-1998-104271-3
【출원인】	
【명칭】	학교법인대우학원
【출원인코드】	2-1999-901351-3
【대리인】	
【성명】	이건주
【대리인코드】	9-1998-000339-8
【포괄위임등록번호】	2003-001449-1
【발명자】	
【성명의 국문표기】	선우명훈
【성명의 영문표기】	SUNWOO,Myung Hoon
【주민등록번호】	580325-1038014
【우편번호】	137-040
【주소】	서울특별시 서초구 반포동 799번지 반포아파트 68-501
【국적】	KR
【우선권주장】	
【출원국명】	KR
【출원종류】	특허
【출원번호】	10-2002-0036216
【출원일자】	2002.06.27
【증명서류】	첨부

【심사청구】 청구

【취지】 특허법 제42조의 규정에 의한 출원, 특허법 제60조의 규정에 의한 출원심사 를 청구합니다. 대리인
이건주 (인)

【수수료】

【기본출원료】	20	면	29,000 원
【가산출원료】	27	면	27,000 원
【우선권주장료】	1	건	26,000 원
【심사청구료】	11	항	461,000 원
【합계】	543,000 원		

【요약서】

【요약】

본 발명은 OFDM(Orthogonal Frequency Division Multiplexing), DMT(Discrete MultiTone) 모뎀의 핵심 기능부에 해당하는 FFT(Fast Fourier Transform) 프로세서에 관한 것으로, 특히 상기 FFT 프로세서는 혼합-기수(Mixed-radix)에 다중 뱅크 메모리를 위한 인-프레이스(In-place) 알고리즘을 적용하여 순차적 입력과 출력을 동시에 수행함으로써 4개의 뱅크로 구성된 2N워드 메모리만으로 연속처리를 구현하여 고속 연산과 동시에 메모리 복잡도를 최소화한 FFT 프로세서를 제공한다.

【대표도】

도 4

【색인어】

FFT, OFDM, 기수, 혼합-기수

【명세서】

【발명의 명칭】

고속 퓨리에 변환을 이용한 혼합-기수 방식의 변조 장치{MODULATING APPARATUS FOR USING FAST FOURIER TRANSFORM OF MIXED-RADIX SCHEME}

【도면의 간단한 설명】

도 1은 종래의 다중 뱅크 구조를 사용하지 않는 혼합-기수(Mixed-radix) 알고리즘 FFT 프로세서의 구조를 나타낸 블록도.

도 2는 종래의 인-프레이스(In-place) 알고리즘을 사용하지 않는 홉합-기수 알고리즘 FFT 프로세서의 구조를 나타낸 블록도.

도 3은 종래의 다중 뱅크 메모리를 위한 기수-4(이하; Radix-4) 인-프레이스 알고리즘을 나타낸 도면.

도 4는 본 발명에 따른 연속 처리 구조를 갖는 In-place 알고리즘 혼합-기FFT 프로세서의 구조를 나타낸 블록도.

도 5는 본 발명에 따른 32-포인트 혼합-기수 FFT 프로세서의 연산 흐름도.

도 6a는 도4의 FFT 프로세서에 사용되는 Radix-4/Radix-2 버터플라이 회로의 블록도.

도 6b는 도6a의 Radix-2 버터플라이 연산 시 등가 버터플라이 쌍을 나타낸 도면.

도 7은 도3의 Radix-4 알고리즘만을 사용할 경우에 연속 처리를 구현하기 위한 FFT 연산 흐름도.

도 8a는 4^n-포인트 FFT 연산을 위해 Radix-4 알고리즘만을 사용할 경우에 출력의 디지트 리버스(Reverse) 순서를 나타낸 도면.

도 8b는2^n(n=3, 5, 7, 9, …)- 포인트 FFT 연산을 위해 Radix-2를 함께 사용하는 Mixed-radix 알고리즘을 사용할 경우에 출력의 비대칭적 리버스 순서를 나타낸 도면.

도 9는 32-포인트 Mixed-radix FFT 연산인 도5의 열4에 해당하는 비대칭적 리버스 출력 순서를 나타낸 도면.

도 10은 32-포인트 Mixed-radix FFT 연산인 도5의 열3에 해당하는 대칭적 리버스 출력 순서를 나타낸 도면.

도 11은 2^n(n=3, 5, 7, 9, …)-포인트 FFT 연산을 위한 혼합-기수의 대칭적 리버스 출력 순서를 나타낸 도면.

도 12는 도5의 열3에 해당하는 대칭적 리버스 출력 순서를 생성하기 위한 데이터 교환을 나타낸 흐름도.

도 13은 32-포인트 혼합-기수 FFT 연산인 도5의 뱅크 인덱스 생성 방법을 나타낸 도면.

도 14는 2^n(n=3, 5, 7, 9, …)-포인트 FFT 연산을 위한 혼합-기수의 뱅크 인덱스 생성 방법을 나타낸 도면.

【발명의 상세한 설명】

【발명의 목적】

【발명이 속하는 기술분야 및 그 분야의 종래기술】

<17> 본 발명은 데이터의 변조 장치에 관한 것으로, 특히 직교 주파수 분할 변조(OFDM : Orthogonal Frequency Division Multiplexing) 방식 또는 이산 다중 톤(DMT : Discrete Multi-Tone) 방식의 변조 장치에 관한 것이다.

<18> 일반적으로 디지털 데이터 통신 시스템에서는 데이터의 송신 및 수신 시에 변조와 복조를 통해 데이터의 전송이 이루어진다. 이러한 변조와 복조는 모뎀(Modem)이라는 장치에서 수행된다. 상기 변조와 복조를 위한 모뎀의 장치는 각 변조 방식에 따라 서로 다른 구성 및 구조를 가지게 된다. 데이터 통신에서 사용되는 일반적인 변조 방식들은 코드 다중 변조(CDM) 방식, 주파수 변조(FDM) 방식, 직교 주파수 변조(이하 'OFDM' 이라 함.) 방식, 이산 다중 톤(이하 'DMT'라 함.) 변조 방식 등이 존재한다.

<19> 그러면 이하에서는 OFDM 방식과 DMT 변조 방식에 대하여 설명한다.

<20> 상기 OFDM 방식은 무선 통신 시스템에서 다중 경로 채널을 통해 고속 데이터 전송을 위해 제안되었다. 상기 OFDM 방식 이전에는 단일 캐리어 전송 방식으로 데이터를 전송하였다. 즉, OFDM 방식 이전의 무선 통신 시스템에서는 전송하고자 하는 직렬의 데이터를 변조하고, 전송 시에 변조된 각 심볼이 전 채널의 주파수 대역을 사용하여 전송하였다. 그러나, OFDM 방식 또는 DMT 방식은 변조된 데이터를 부반송파(subcarrier)의 수만큼 직·병렬 변환하고 이를 각각에 대응되는 부반송파(subcarrier)로 변조하는 방식이다. 이러한 부반송파를 이용한 변조는 이산 퓨리에 변환(Discrete Fourier Transform : 이하 'DFT'라 함)을 이용하여 구현한다.

그러나, 실제 하드웨어 설계에는 DFT나 역 이산 퓨리에 변환(Inverse Discrete Fourier Transform : 이하 'IDFT'라 함)을 사용하지 않고 연산량을 줄이기 위해 고속 퓨리에 변환(Fast Fourier Transform : 이하 'FFT'라 함) 알고리즘을 이용하여 구현한다. 이러한 FFT 알고리즘을 처리하기 위한 프로세서는 OFDM 시스템에 있어 가장 큰 복잡도를 가지며 고속 연산이 요구된다. 따라서 FFT 알고리즘을 처리하기 위한 프로세서는 구현이 까다로운 부분이다.

<21> 이러한 고성능을 요구하는 분야에 사용되는 FFT 프로세서는 주로 파이프라인(pipe line) 구조를 가지는 FFT 프로세서가 사용된다. 그러나, 이 구조는 스테이지(stage) 수만큼의 연산부를 요구하여 포인트 수가 증가할 경우 실제 하드웨어로 구현할 경우에 많은 면적을 소모한다. 따라서 상기한 문제점을 극복하고 작은 하드웨어 크기를 유지하기 위해 메모리 구조와 단일 버터플라이 연산부를 사용하는 프로세서들이 발표되고 있다.

<22> 그 한 예로서 기수-2(Radix-2 : 이하 'Radix-2'라 함) FFT 알고리즘을 사용한 메모리 구조 FFT 프로세서가 있다. 상기 메모리 구조 FFT 프로세서는 메모리 구조에 상기 Radix-2 알고리즘을 사용할 수 있으므로, 곱셈기의 수를 최소화할 수 있다. 따라서 메모리 구조 FFT 프로세서는 작은 면적을 소모하는 FFT 프로세서의 구현이 가능하다.

<23> 그러나 상기 Radix-2 알고리즘을 사용한 메모리 구조 FFT 프로세서는 매우 많은 연산 싸이클을 필요로 한다. 따라서 연산 시간이 매우 길어지는 문제를 가진다. 따라서 고속 연산이 요구되는 OFDM 시스템 또는 DMT 시스템에 적합하지 않으며 이를 만족시키기 위해서는 매우 높은 동작 주파수가 요구된다. 따라서 일반적으로 OFDM 시스템 또는 DMT 시스템에서는 Radix-2 방식보다는 기수-4(Radix-4 : 이하 'Radix-4'라 함) 알고리즘을 사용하는 방식이 주로 이용하고 있다. 그러면 이하에서 현재까지 발표되어 사용되고 있는 Radix-4 알고리즘 방식의 FFT 프로세서들에 대하여 살펴본다.

<24> 도 1은 암피온(AMPHION) 사에서 발표한 Radix-4 알고리즘을 기반으로 한 FFT 프로세서를 나타낸 것이다. 상기 Radix-4 알고리즘의 경우 Radix-2 알고리즘에 비해 스테이지 수가 1/2로 줄어들게 된다. 또한 각 스테이지 당 버터플라이 연산 횟수도 Radix-2에 비해 1/2이 된다. 따라서 Radix-4 알고리즘은 Radix-2 알고리즘에 비해 훨씬 적은 버터플라이 연산 횟수를 가진다. 이와 같은 Radix-4 알고리즘과 Radix-2 알고리즘 및 후술할 혼-기수 알고리즘의 FFT 길이에 따른 연산 횟수를 하기 <표 1>에 도시하였다.

<25> 【표 1】

FFT 길이	Radix-2	Radix-4	혼합-기수
256	1,024	256	-
512	2,304	-	640
1,024	5,120	1,280	-
2,048	11,264	-	3,072
4,096	24,576	6,144	-
8,192	53,248	-	14,336

<26> 상기 <표 1>에서 도시한 바와 같이 Radix-4 알고리즘은 4^n(이하에서 n은 정수를 의미한다)의 길이를 갖는 FFT 연산만이 가능하며, Radix-2 알고리즘은 2^n의 길이를 갖는 모든 FFT의 길이에 대하여 연산이 가능하다. 이를 상기 <표 1>에 있는 FFT 길이를 이용하여 설명하면 하기와 같다. FFT 길이 256은 2^8이므로 Radix-2 알고리즘과 Radix-4 알고리즘 모두에서 사용이 가능하다. 그러나 FFT 길이 512는 2^9이므로 Radix-4 알고리즘은 FFT 길이 512인 경우에 연산을 수행할 수 없다. 따라서 2^n의 길이를 갖는 모든 FFT 연산을 가능하도록 하기 위해 Radix-4 알고리즘을 다른 Radix와 함께 사용하는 혼합-기수(Mixed-radix) 알고리즘이 요구된다. 상기 <표 1>의 마지막 열에는 Radix-4 알고리즘과 Radix-2 알고리즘을 사용하는 Mixed-radix 알고리즘을 사용하였을 경우의 버터플라이 연산 횟수를 나타내었다. 상기 <표 1>의 혼합-기수에 대한

연산 횟수는 상기 도 1의 암피온 사(社)에서 제공하는 장치에서 수행되는 연산 횟수가 된다. 그러면 도 1을 참조하여 암피온 사에서 제공하는 장치에 대하여 살펴본다.

<27> 상기 혼합-기수 알고리즘을 사용하는 상기 도 1의 FFT 프로세서는 Radix-4 버터플라이와 Radix-4/Radix-2의 버퍼플라이를 선택적으로 연동하여 Radix-4, Radix-8, Radix-16의 혼합-기수 연산을 수행한다.

<28> 입출력 인터페이스 및 제어기(11)는 외부로부터 입력되는 데이터(X)를 FFT 연산하도록 제어하며, FFT 연산이 수행된 결과 데이터(Y)를 프로세서 외부로 출력한다. 여기서 입출력 인터페이스 및 제어기(11)로 입력되는 데이터(X)와 출력되는 데이터(Y)는 OFDM 심볼 또는 DMT 심볼이 될 수 있다. 메모리 제어기(12)는 상기 입출력 인터페이스 및 제어기(11)로부터 FFT 연산을 위해 입력되는 데이터와 연산 중인 데이터의 읽기 및 쓰기를 위해 메모리(13)의 주소생성을 제어한다. 상기 메모리(13)는 1024-워드 듀얼 포트 메모리로 구성되어 있으며, 외부로부터 입력되는 데이터 및 FFT 연산 중간의 데이터, 결과 데이터를 상기 메모리 제어기(12)가 지시하는 주소에 저장하거나 독취한다.

<29> 또한 버터플라이 연산부(10)는 Radix-4 버터플라이(14)와 회전인자 룩업 테이블(Look Up Table : 이하 'LUT'라 함)(16)과, 복호 곱셈기(15)로 구성된다. 상기 Radix-4 버터플라이(14)는 Radix-4 버터플라이 연산 중 덧셈과 뺄셈의 연산을 수행한다. 회전인자 LUT(16)는 연산이 수행되는 데이터의 회전인자를 저장하며, 회전인자 값을 출력하는 메모리 테이블이다. 복소 곱셈기(15)는 Radix-4 버터플라이 연산 중 복소수에 대한 곱셈을 수행하고 그 결과 값을 출력한다. 또한 Radix-4/Radix-2 선택적 버터플라이(17)는 FFT 길이에 따라 필요한 최종 연산을 선택적으로 수행하기 위한 장치이다. 즉, FFT 길이에 따라 최종 연산이 Radix-2 연산을 필요로 하는 경우에는 Radix-2 버터플라이를 선택하여 Radix-2 연산을 수행하도록 하며, 최종 연산이 Radix-4

연산을 필요로 하는 경우 Radix-4 버터플라이를 선택하여 Radix-4 연산을 수행하도록 한다. 따라서 전체적인 FFT 연산이 연산부(10)의 Radix-4 버터플라이 연산과 연동하여 Radix-8 혹은 Radix-16 연산을 수행 가능토록 한다. 그러므로 마지막 스테이지에서만 Radix-4/Radix-2 선택적 버터플라이(17)를 사용하고 나머지 스테이지에서는 Radix-4 버터플라이 연산부(10)만을 사용하기 위한 MUX(18)로 구성된다. 상기 Radix-4 알고리즘의 경우 4개의 입력과 출력을 갖는 버터플라이로 구현된다. 따라서 4개의 입출력이 한 싸이클에 수행되어야 연산 싸이클을 최소화할 수 있다. 이와 같이 한 싸이클에 4개의 입출력이 이루어지기 위해서는 메모리를 다중 뱅크로 나누어 사용하여야만 한다. 그러나, 상기 도 1의 FFT 프로세서는 다중 뱅크 구조를 사용하지 않는 구조로 되어있다. 따라서 도 1과 같은 구조로 동작을 수행하는 경우 많은 연산 싸이클이 요구되어 Radix-4 연산의 장점을 살리지 못하고 있다.

<30> 도 2는 드레이 엔터프라이즈(Drey Enterprise) 사에서 발표한 혼합-기수 알고리즘과 다중 뱅크 구조를 갖는 FFT 프로세서의 블록 구성도이다. 상기 도 2에 도시한 바와 같이 드레이 엔터프라이즈 사에서 발표한 FFT 프로세서도 메모리 구조를 가짐을 알 수 있다. 도 2의 FFT 프로세서에서는 두 개의 메모리들(RAM)(21, 22) 중 하나의 RAM이 외부 입력 데이터를 저장하는 동안 나머지 하나의 RAM은 FFT 연산에 사용된다. 그리고, MUX(23)는 입력 RAM들 중 하나로부터 버터플라이 입력을 받을지 출력 RAM들(28, 29) 중 하나로부터 버터플라이 입력을 받을지를 결정한다. Radix-2 연산부들(26, 27)은 Radix-2 연산을 수행하는 스테이지에서 Radix-2 연산을 수행하여 그 결과를 출력한다. 또한 MUX(24)는 Radix-2 연산부들(26,27)로부터 Radix-2 버터플라이 출력을 받아 입력 RAM들(21, 22) 중 하나에 저장하거나 또는 출력 RAM들(28, 29) 중 하나에 저장하기 위해 결과 값을 다중화하여 출력한다. Radix-2/Radix-4 공유연산부(25)는 Radix-4 연산을 수행하는 스테이지에서는 Radix-4 연산을 수행하고, Radix-2 연산을 수행하는

스테이지에서는 Radix-2 연산을 수행하여 그 결과 값을 출력한다. 2개의 출력 RAM들(28, 29) 중 하나의 RAM이 FFT 연산에 사용되는 동안 나머지 하나의 RAM은 FFT 연산의 결과 데이터를 외부로 출력한다. 상기 도 2와 같은 구조는 Radix-4와 Radix-2의 혼합-기수 알고리즘을 사용하며 또한 메모리도 다중 뱅크 구조를 사용한다. 따라서 메모리의 다중 뱅크 구조를 사용함으로 연산 클럭 싸이클이 최소화된 구조이다.

<31> 그러나, 상기 도 2의 구조는 버터플라이 입력을 억세스한 메모리 위치에 버터플라이 출력을 저장하는 인-플레이스(In-plase) 알고리즘을 적용하지 못하였다. 따라서 FFT 연산에 N워드를 갖는 메모리 2개를 사용하는 구조이다. 즉, 상기 FFT 연산만을 위해서는 각각 4개의 뱅크로 구성된 메모리들 중 2개의 메모리만으로도 가능하지만, 연속 처리를 수행하기 위해서는 입력과 출력을 위한 2개의 메모리를 더 사용해야만 한다. 따라서, 상기 도 2에서는 총 4개의 메모리가 사용된다. 메모리의 경우 FFT 프로세서에서 대부분의 면적을 차지하는 블록 중에 하나이다. 따라서 상기와 같이 메모리의 숫자가 증가되는 경우에 메모리의 복잡도가 증가하며, 하드웨어의 부피가 증대되고, 가격이 상승하는 요인으로 작용한다.

<32> 상기 메모리 구조들의 메모리 복잡도를 최소화하기 위해 엘.쥐. 존슨(L. G. Johnson)이 발표한 인-플레이스 알고리즘의 16-포인트 FFT에 대한 실시 예는 도 3과 같다. 상기 인-플레이스 알고리즘은 메모리를 다중 뱅크로 나누어 사용할 경우를 위한 알고리즘이다. Radix-4 버터플라이 연산을 위해 4개의 데이터를 동시에 억세스하여야 하고, 버터플라이 연산이 이루어진 4개의 결과를 억세스한 위치에 동시에 저장하여야 한다. 이를 위해 메인 메모리를 4개의 뱅크(뱅크0, 뱅크1, 뱅크2,

뱅크3)로 나누어 사용하며 동시에 한 뱅크에서 여러 개의 데이터를 억세스하지 않기 위해 적절한 어드레싱을 수행해야만 한다. 도 3은 16-포인트 FFT에 대한 인-플레이스 메모리 어드레싱의 예이다. 상기 도 3을 살펴보면, 첫 번째 버터플라이 연산부터 8번째의 버터플라이 연산까지를 수행하는 구성을 가진다. 또한 각 버터플라이 연산들을 살펴보면, 1회에 4개씩의 입력을 취한다. 이때 4개의 입력들은 각기 다른 뱅크들로부터 읽어온다. 그러면 첫 번째 버터플라이 연산과 두 번째 버터플라이 연산에 대하여 살펴본다. ①번 버터플라이의 경우 4개의 입력을 뱅크의 0번지, 뱅크1의 1번지, 뱅크2의 2번지, 뱅크3의 3번지로부터 읽어오고, 버터플라이 연산 결과를 같은 뱅크들과 번지들에 저장한다. ②번 버터플라이의 경우 4개의 입력을 뱅크1의 0번지, 뱅크2의 1번지, 뱅크3의 2번지, 뱅크0의 3번지로부터 읽어오고, 버터플라이 연산 결과를 같은 위치에 저장한다. 도 3에서 어느 뱅크를 사용하는지를 나타내는 뱅크 인덱스(뱅크i)는 데이터 입력 카운트 비트들을 2비트 디지트들로 나누어 모듈로-4 덧셈을 취함으로써 쉽게 구할 수 있다. 도 3이 16-포인트 FFT이므로 16개의 데이터를 카운트하기 위해서 4 비트 카운터가 사용된다. 4 비트를 상위 2 비트와 하위 2 비트의 디지트로 나누고, 상위 2 비트와 하위 2 비트를 모듈로-4 덧셈을 수행하는 방식으로 뱅크 인덱스를 구한다.

<33> 그러나, 이상에서 상술한 인-플레이스 알고리즘은 혼합-기수가 아닌 고정된 기수에만 사용하기 위해 제안되었다. 따라서 상기한 인-플레이스 알고리즘을 혼합-기수 방식에 그대로 적용할 수 없는 문제가 있다.

<34> 다음은 연속 처리 구조를 갖는 종래의 구조에 대해 기술한다. 메모리 구조에

서 입력과 출력을 동시에 수행함으로써 N 워드의 크기를 갖는 메모리 2개만을 사용하여 연속처리가 가능한 메모리 구조의 FFT 프로세서가 알.랜드후앤(R. Radhouane)에 의해 제안되었다. 상기 구조는 Radix-2 알고리즘에서 DIF 연산을 할 경우와 DIT 연산을 할 경우에 각각 출력과 입력이 비트 리버스 특성을 갖는 것을 이용하여 한번은 DIF 연산을 수행하고 다음은 DIT 연산을 수행하는 방식으로 연속 처리를 구현하였다. 하기의 <표 2>는 메모리 2개를 이용한 연속 처리 구조의 연산 방식을 도시하였다.

<35> 【표 2】

	메모리 1		메모리 2	
OFDM 심볼	메모리 상태	FFT-I/O 모드	메모리 상태	FFT-I/O 모드
0	C	DIF	I/O	NAT
1	I/O	BR	C	DIF
2	C	DIT	I/O	BR
3	I/O	NAT	C	DIT

<36> 상기 <표 2>에서 OFDM 심볼은 FFT 연산의 길이에 해당하는 데이터를 의미한다. 예를 들어 256-포인트 FFT 연산의 경우 하나의 OFDM 심볼은 256개의 데이터들을 의미한다. 상기 <표 2>에서 C는 FFT 연산을 의미하고, I/O는 입출력을 수행함을 의미한다. NAT는 원래의 0, 1, 2, 3, …, N-1 번지의 올바른 순서로 메모리 어드레싱을 수행하여 입출력하는 것을 의미하며, BR은 비트 리버스 어드레싱으로 메모리 입출력을 수행함을 의미한다. 또한 상기 <표 2>의 0번 OFDM 심볼에서 메모리1이 DIF로 연산을 수행하는 동안 메모리2는 NAT 즉, 올바른 순서로 어드레싱을 수행하여 입출력을 수행한다. 다음 1번 OFDM 심볼에서 메모리1은 BR 즉, 비트 리버스 어드레싱으로 입출력을 수행하고, 메모리2는 DIF로 연산을 수행한다. 2번 심볼에서는 메모리1이 DIT로 연산을 수행하고, 메모리2는 BR 즉, 비트 리버스 어드레싱으로 입출력을 수행한다. 다음 3번 심볼에서는 메모리1이 NAT 즉, 올바른 순서로 입출력을 수행하고 메모리2는 DIT로 연

산을 수행한다. 다음 4번 심볼부터는 0, 1, 2, 3번 심볼의 연산흐름이 반복된다. 2개의 메모리로 연속처리가 가능하려면 하나의 메모리가 연산을 수행하는 동안 나머지 메모리는 순차적 순서를 갖는 입력과 출력을 동시에 수행 가능하여야 한다. 이와 같이 상기 구조에서는 두 메모리가 입출력과 FFT 연산을 번갈아 가며 수행하는 방식으로 2개의 메모리만으로 연속 처리가 가능하였다.

<37> 알카텔(Alcatel) 사에서 발표한 종래의 구조가 3개의 메모리를 사용하여 연속 처리를 구현한 반면 상기 종래의 연속 처리 구조는 2개의 메모리만을 사용하여 메모리 복잡도를 최소화할 수 있었다.

<38> 그러나, 상기 연속 처리 구조는 Radix-2 알고리즘을 사용하는 경우만을 위해 구성되었다. 따라서 Radix-2 연산만을 수행하므로 많은 연산 싸이클과 높은 동작 주파수를 요구하는 문제점이 있었다.

【발명이 이루고자 하는 기술적 과제】

<39> 상술한 문제점을 해결하기 위하여 본 발명은 FFT 프로세서에 있어서, 고속 연산과 동시에 최소화된 복잡도를 갖는 회로를 제공함으로써 고성능을 만족시키며 면적을 최소화할 수 있는 프로세서를 제공하는 것을 목적으로 한다.

<40> 본 발명의 다른 목적은 FFT 연산 수행 시에 Radix-2/Radix-4의 혼합-기수를 제공하면서 작은 면적과 복잡도를 줄일 수 있는 프로세서 장치를 제공함에 있다.

<41> 본 발명의 또 다른 목적은 FFT 연산을 수행하는 프로세서에서 Radix-2/Radix-4의 혼합-기수를 제공하면서 빠른 속도의 처리가 가능한 프로세서 장치를 제공함에 있다.

<42> 본 발명의 또 다른 목적은 FFT 연산을 수행하는 프로세서에서 Radix-2/Radix-4의 혼합-기수를 제공하면서 연속 처리가 가능한 프로세서 장치를 제공함에 있다.

<43> 상기 목적들을 달성하기 위하여 본 발명은 Mixed-radix 알고리즘과 연속처리 구조를 갖는 FFT 프로세서에 있어서, 선택한 메모리와 그 선택한 메모리의 임의의 뱅크로부터 입력과 출력을 수행하기 위한 입출력 인터페이스와; 상기 인터페이스로의 입출력과 FFT 연산에 사용되는 4개의 뱅크로 구성된 2개의 N 워드 메모리와; 상기 선택한 메모리와 뱅크로부터 4개의 버플라이 입력을 공급하기 위한 데이터 교환부와; 상기 데이터 교환부로부터 공급된 Radix-4와 Radix-2 두 종류의 버터플라이를 하나의 회로로 수행하며 대칭적인 리버스 출력을 구성하기 위한 Radix-4/2 버터플라이 연산 회로와; 상기 선택한 메모리와 뱅크로 4개의 버터플라이 출력을 공급하기 위한 데이터 교환부와; 다중 뱅크 메모리 구조에서 인-플레이스 연산을 수행하기 위한 주소를 생성하는 주소 생성부를 포함하는 FFT 프로세서를 특징으로 한다.

<44> 상기한 목적들을 달성하기 위한 본 발명은 고속 퓨리에 변환을 이용한 혼합-기수 방식의 변조 장치로서, 입력 심볼을 저장하거나 고속 퓨리에 연산이 완료된 심볼들을 저장하는 4개의 뱅크로 구성된 2개의 메모리들과, 상기 메모리로부터 출력되는 심볼들의 수에 따라 기수-4 또는 기수-2 방식으로 버터플라이 연산을 수행하고, 상기 연산된 값들을 대칭적 리버스로 출력하는 버터플라이와, 상기 메모리들 중 하나의 메모리의 각 뱅크들로부터 각각 하나씩의 심볼들을 독취하여 상기 버터플라이로 출력하는 제1데이터 교환부와, 상기 버터플라이로부터 연산되어 출력된 심볼들을 상기 제1데이터 교환부에서 읽어온 주소와 동일한 주소에 저장되도록 각 심볼들을 매칭하는 제2데이터 교환부와, 상기 제1데이터 교환부에서 읽어온 심볼이 연산 이후 상기 제2데이터 교환부를 통해 출력될 시 상기 제1데이터 교환부에서 읽어온 뱅크 및 주소와 상기

제2데이터 교환부의 출력 뱅크 및 주소가 동일하도록 상기 제2데이터 교환부의 출력을 제어하는 주소 생성부를 포함함을 특징으로 한다.

【발명의 구성 및 작용】

<45> 이하, 첨부된 도면을 참조로 하여 본 발명을 상세히 설명하기로 한다.

<46> 도 4는 본 발명에 따른 인-플레이스 알고리즘을 적용하고 Mixed-radix 구조와 입출력을 동시에 수행함으로써 2N 워드의 메모리로 연속 처리를 수행하는 FFT 프로세서의 블록 구성도이다.

<47> 입출력 인터페이스(101)는 입력되는 데이터를 FFT 연산을 위해 메모리들(102, 103) 중 한 메모리를 선택하고, 메모리의 4개의 뱅크 중 하나의 뱅크를 선택하여 데이터를 기록한다. 그리고 입력된 데이터의 연산이 완료된 경우 상기 입출력 인터페이스(101)는 FFT 연산이 완료된 데이터가 저장된 메모리를 선택하고, 상기 선택된 메모리의 4개의 뱅크 중 하나의 뱅크를 선택하여 데이터를 독출하여 출력한다. 4개의 뱅크를 가지는 2개의 N 워드 메모리들(102, 103)는 상기 인터페이스(101)로부터 입력되는 데이터를 저장하고, 저장된 데이터의 FFT 연산을 위해 제1데이터 교환부(104)로 출력하고, 제2데이터 교환부(106)로부터 수신되는 데이터를 다시 저장한다. 그리고, 상기 각 메모리들(102, 103)에서 FFT 연산이 완료된 데이터는 입출력 인터페이스(101)로 출력한다. 상기 제1데이터 교환부(104)는 FFT 연산을 수행할 데이터가 저장된 메모리 및 상기 메모리의 뱅크를 선택하고, 본 발명에 따라 인-플레이스 연산을 위해 선택된 메모리의 뱅크로부터 4개의 데이터를 읽어온다. 그리고, 상기 제1데이터 교환부(104)는 주소 생성부(107)로부터 출력되는 주소 생성 값에 따라 상기 읽어온 데이터를 교환하여 Radix-4/2

버터플라이(105)로 출력한다. 또한 본 발명에 따른 Radix-4/2 버터플라이(105)는 상기 제1데이터 교환부(104)로부터 입력되는 데이터에 따라 Radix-4 방식으로 동작하거나 또는 Radix-2 방식 중 하나의 방식으로 연산을 수행한다. 상기 Radix-4/2 버터플라이(105)는 하나의 회로로 이루어지며, 대칭적인 리버스 출력을 가진다. 이와 같이 Radix-4/2 버터플라이(105)에서 연산된 데이터는 제2데이터 교환부(106)로 출력된다. 상기 제2데이터 교환부(106)는 상기 주소 생성부(107)로부터 수신되는 주소에 따라 상기 Radix-4/2 버터플라이(105)로부터 출력된 값을 저장할 메모리와 상기 메모리 내의 뱅크를 선택하여 저장한다. 주소 생성부(107)는 상기 다중 뱅크 메모리 구조에서 본 발명에 따른 인-플레이스 연산을 수행하기 위한 뱅크 인덱스 및 주소를 생성하여 출력한다.

<48> 상기 도 4의 구조를 가지는 FFT 프로세서는 도 5와 같은 흐름도를 갖는다. 도 5는 32-포인트 Mixed-radix FFT 연산의 예이며, 스테이지1과 스테이지2는 Radix-4로, 마지막 스테이지3은 Radix-2로 연산이 수행된다. 상기 도 4에서 살핀 바와 같이 본 발명의 구조에서는 2개의 각 메모리를 4개의 뱅크로 나누어 동시에 4개의 데이터 접근이 가능하다. 그러므로 2개의 데이터를 사용하는 Raidx-2 버터플라이의 경우 동시에 2개의 버터플라이 연산 수행이 가능하다. 또한 스테이지3의 가는 실선 안에 표시한 2개의 Radix-2 버터플라이들이 동시에 수행 가능한 버터플라이 쌍을 나타낸다. 이와 같이 2개의 Radix-2 버터플라이를 동시에 한 싸이클로 수행하면 연산 싸이클 감소의 이득을 볼 수 있다.

<49> 상기 FFT 프로세서에서 Radix-2 버터플라이 구조는 별도의 버터플라이로 구현하지 않고 Radix-4 버터플라이에 데이터 스위칭 회로를 추가하여 구현한다. 이를 도 6a에 나타내었다. 도 6a에서 'Radix-2'라는 멀티플렉서(Multiplexer) 선택 신호를 통해 Radix-4 버터플라이와 2개의 Radix-2 버터플라이를 구현한다. 상기 도 6a에 도시한 도면에 대하여 좀더 살펴보면 하기와

같다. 2개씩 입력되는 데이터 x(0)의 심볼과 x(2)의 심볼이 각 Radix-2 버터플라이를 구성하는 가산기들로 각각 입력되며, x(1)의 심볼과 x(3)의 심볼이 다른 Radix-2 버터플라이를 구성하는 가산기들로 각각 입력된다. 이와 같이 각 Radix-2 버터플라이를 구성하는 가산기들은 출력을 둘로 분기하여 하나의 출력을 각각 멀티플렉서의 한 입력단으로 입력된다. 그리고, 상기 분기된 다른 하나의 출력은 Radix-2 버터플라이를 구성하는 각기 다른 가산기로로부터의 출력과 가산 및 감산을 수행하여 상기 각 멀티플렉서들의 다른 한 입력단으로 입력된다. 이후 각 멀티플렉서들의 출력은 도면에 도시한 바와 같이 다시 분기되어 다른 멀티플렉서들을 통해 선택이 이루어지거나 또는 그대로 출력된다. 이와 같이 출력된 값들은 본 발명에 따라 인-플레이스를 위한 주소로 매핑된다. 이에 대하여는 이하에 첨부된 도면을 참조하여 더 상세히 설명하기로 한다. 상기 도 6a에 대한 등가 회로를 구성하면 도 6b와 같이 구성된다. 도 6b는 Radix-4 버터플라이 회로로 Radix-2 버터플라이를 구현하였을 경우 등가의 Radix-2 버터플라이 쌍을 나타낸다.

<50> 다음으로 상기 본 발명의 메모리 구조에서 연속 처리를 수행하기 위한 구조에 대해 설명한다. 종래기술에서 설명한 바와 같이 알.래드후앤에 의해 제안된 연속 처리 구조는 Radix-2 알고리즘을 위한 구조이다. 그러나, 본 발명의 구조는 Radix-4와 혼합-기수를 위한 구조이다. 또한, 종래의 구조는 FFT와 DIT 연산을 번갈아 가며 수행하는 방식이다. 그러나, 본 발명의 구조는 DIF 연산만을 수행하며 메모리 어드레싱의 제어만으로 연속 처리를 수행한다.

<51> 본 발명에서와 같이 연속 처리를 수행하기 위해서는 복호를 수행할 심볼의 위치에 새로이 복호할 심볼이 저장되어야 한다. 이와 같이 복호가 될 심볼을 읽어가는 동작과 새로이 복호할 심볼이 저장되는 것은 한 클럭에 의해 읽기와 쓰기가 동시에 이루어진다. 또한 Radix-4 버터플라이를 만족하기 위해서는 4개의 뱅크를 가지는 메모리에서 각 뱅크에서 출력되는 심볼이

하나씩만 출력되어야 한다. 그리고, 각 스테이지를 거쳐 연산이 완료되었을 때, 상기 심볼들은 읽혀진 위치에 기록되어야만 한다. 따라서 이러한 방법으로 주소를 생성하는 것이 가장 큰 과제가 된다. 이상에서 설명한 방법은 Radix-4 알고리즘만을 사용할 때 쉽게 구현될 수 있다. 따라서 도 7에는 Radix-4 알고리즘만을 사용하는 16-포인트 FFT의 예를 도시하였다.

<52> 도 7에서 열1과 열2는 데이터들이 저장되는 메모리 뱅크와 번지를 나타낸 것이고, 열3은 디지트 리버스 출력 순서를 나타낸다. 도 7에서 처음으로 외부에서 들어온 입력 데이터들은 열2의 뱅크와 번지에 저장된다. 상기 입력된 데이터들은 첫 번째 스테이지에서 4번의 버터플라이 연산이 이루어지며, 2번째 스테이지에서 4번의 버터플라이 연산이 이루어진다. 즉, 도 7에서 ① 내지 ④는 첫 번째 스테이지에서 이루어지는 4번의 버터플라이 연산을 나타내며, ⑤ 내지 ⑧은 두 번째 스테이지에서 이루어지는 4번의 버터플라이 연산을 나타낸다. 상기 각 스테이지의 각 버터플라이 연산은 하나의 메모리에 구비된 4개의 각 뱅크로부터 데이터가 하나씩 읽혀짐을 알 수 있다. 첫 번째 버터플라이 연산에서는 뱅크0의 0번지 데이터와, 뱅크1의 1번지 데이터와 뱅크2의 2번지 데이터와 뱅크3의 3번지 데이터가 읽혀진다. 따라서 각 뱅크마다 하나씩 데이터가 읽혀지면서 처리가 이루어진다. 이러한 방법으로 4번의 버터플라이 연산이 완료되면, 2번째 스테이지에서 다시 4번의 버터플라이 연산이 이루어진다.

<53> 상기한 방법으로 연산이 이루어진 출력들은 도 7의 열3에 도시한 바와 같은 출력 순서를 가진다.

<54> 상술한 바와 같이 FFT 연산이 수행되면 열3의 디지트 리버스 순서로 출력이 구성되며 메모리 저장 위치는 인-플레이스 연산을 수행하므로 열2의 뱅크와 번지가

그대로 유지된다. 연산 결과를 디지트 리버스 어드레싱을 통해 순차적 순서로 출력하고 동시에 순차적 순서로 다음 데이터를 입력하면 열1의 뱅크와 주소에 새로운 입력 데이터들이 저장된다. 열3의 '0'번 출력은 열2의 뱅크0의 0번지에 저장되어 있으므로 이를 출력하고, 다음 FFT 연산을 위한 새로운 데이터의 '0'번을 상기 출력이 이루어진 뱅크와 번지에 저장한다. 다음으로 열3의 '1'번 출력은 열2의 뱅크1의 1번지에 있으며, 이를 출력하고 새로운 데이터의 '1'번을 상기 출력이 이루어진 위치에 저장한다. '2'번 출력은 열2의 뱅크2의 2번지에 저장되어 있으며 이를 출력하고 이 위치에 새로운 데이터의 '2'번을 상기 출력이 이루어진 위치에 저장한다. 이와 같은 방법으로 새로운 입력을 저장하면 열1의 뱅크와 번지가 형성된다. 그리고 이에 대한 FFT 연산을 수행한 뒤 순차적 순서로 출력하고 순차적 순서로 입력을 저장하면 다시 열2의 뱅크와 주소 할당으로 복원된다. 따라서 열1과 열2의 뱅크와 주소 할당이 번갈아 가며 수행된다.

<55> 이와 같이 순차적 순서로 동시에 입출력 수행이 가능하면 하나의 메모리가 연산을 수행하는 동안 나머지 메모리는 입출력을 수행하는 방식으로 메모리 2개만으로 연속 처리가 가능하다. 이때 FFT 연산은 입출력 동작 주파수에 비해 2배의 동작 주파수로 연산이 수행되어야 한다. 이는 상기 <표 1>에서 볼 수 있듯이 FFT 연산 클럭 싸이클이 FFT 포인트 수보다 더 길기 때문이다. 즉, Radix-4의 경우 1024-포인트 FFT부터, Radix-4/Radix-2의 혼합-기수의 경우 512-포인트 FFT부터 연산 클럭 싸이클이 더 길어진다.

<56> 혼합-기수의 경우 연속 처리를 위한 순차적 입출력을 위해 별도의 조작이 필요하다. 32-포인트 혼합-기수의 경우 출력이 상기 도 5의 열4와 같이 나타난다. 이는 도 7 열3의 Radix-4 알고리즘의 리버스 순서와 달리 비대칭적인 리버스 형태를 갖는다. 먼저 Radix-4 알고리즘만을 사용한 경우의 디지트 리버스 순서에 대해 설명하면, Radix-4의 2^n-포인트 FFT에 대한 디지트

리버스 순서를 도 8a에 나타내었다. 2^n개의 데이터를 카운트하기 위해서 n 비트가 필요하므로 n 비트 카운터가 사용된다. 도 8a에서 (n-1, n-2) 번째 비트, (n-3, n-4) 번째 비트, …, (3, 2) 번째 비트, (1, 0) 번째 비트들의 쌍을 하나의 디지트로 하여 리버스가 수행된다. 본 발명에서는 이와 같이 이루어지는 리버스를 '디지트 리버스'라 칭함에 유의하여야 한다. 도 8a에서 알 수 있듯이 디지트들의 정가운데를 중심으로 대칭적으로 리버스가 됨을 알 수 있다.

<57> 도 8b는 2^n-포인트 FFT에 대한 혼합-기수의 리버스 순서를 도시하였다. 혼합-기수의 경우 23, 25, 27, 29 등의 2의 홀수승의 포인트 수를 가져 출력 카운트 비트의 수가3, 5, 7, 9 비트 등과 같이 홀수이기 때문에 Radix-4와 같이 2 비트 디지트로만 리버스할 수 없다. 최하위 비트가 별도로 리버스 되어야 하며 이로 인해 비대칭적인 리버스 형태를 갖게 된다. 도 8b의 예로 32(2^5)-포인트에 해당하는 도 5의 열4는 도 9와 같은 비대칭적 리버스 형태를 가진다. 비대칭적인 리버스 출력을 가질 경우 Radix-4 알고리즘으로 구성된 도 7과 같이 열1과 열2의 뱅크와 번지가 반복되는 구조를 구성할 수 없다. 혼합-기수에서도 Radix-4만을 사용하는 도 7과 같이 연속 처리를 위해 열1과 열2가 반복되는 구조를 가지려면 출력이 대칭적인 리버스 형태를 가져야 한다. 이를 위해 혼합-기수 알고리즘에서 도 5의 열4와 같은 비대칭적 출력 순서를 열3과 같은 대칭적 출력 순서를 갖도록 변환하는 데이터 교환이 수행되며 이를 통한 32(2^5)-포인트의 대칭적 리버스 출력 순서가 도 5의 열3과 같으며 도 10과 같이 생성할 수 있다. 이를 일반화하여 2^n-포인트 혼합-기수 FFT의 경우를 살펴보면 출력의 대칭적 리버스 순서는 도 11과 같으며 상위 2 비트(n-1, n-2)와 하위 2비트 (1, 0)는 디지트 리버스가 수행되며 가운데 비트들(n-3, n-4, …, 3, 2)은 비트 리버스가 수행된다. 결론적으로 원래 도 8b의 비대칭적 리버스 형태가 데이터 교환 과정을 통해 도 11의 대칭적 리버스 형태로 변환되는 것이다.

<58> 도 5의 열3과 도 10에 나타낸 32-포인트의 대칭적 리버스 순서를 구현하기 위한 데이터 교환 과정을 도 12에 도시하였다. 상기 도 12는 상기 도 5의 우측 상단에 굵은 실선으로 박스(box) 표시한 8-포인트 DFT 부분에 해당한다. 도 12에 나타낸 바와 같이 단순히 스테이지2에서는 Radix-4 버터플라이의 두 번째와 세 번째의 각 2개씩 존재하는 출력 g'2(n), g'3(n)의 저장 위치를 교환하고, 스테이지3에서도 Radix-2 버터플라이 쌍의 출력 X'2(n)과 X'3(n)의 저장 위치를 교환한다. 이와 같이 구성하면 도 12의 열1에 도시한 바와 같이 대칭적 리버스 순서로 변환된다. 상기 도 12에서 열2는 본 발명에서와 같은 저장 위치의 교환이 이루어지지 않은 경우의 비대칭적 리버스 순서를 대비하여 도시하였다. 이와 같은 데이터 교환은 도 6a의 버터플라이 회로에서 'Exchange' 신호의 제어를 통해 수행될 수 있다. 예로 든 32-포인트 FFT 뿐만 아니라 모든 2^n(n=1, 3, 5, 7, 9, …)-포인트 FFT에서 첫 번째 스테이지를 제외한 나머지 스테이지에서 Radix-4 버터플라이의 두 번째와 세 번째 출력 저장 위치를 교환하고 마지막 스테이지에서 2개의 Radix-2 버터플라이의 두 번째와 세 번째 출력 저장 위치를 교환하는 것에 의해 대칭적 리버스 출력 순서를 구성하는 것이 가능하다.

<59> 마지막으로 Mixed-radix 알고리즘에서의 뱅크 인덱스 생성에 대해 설명한다. 본 발명에 따른 뱅크 인덱스 생성 방법은 전술한 도 5의 열1과 열2에 나타낸 뱅크 인덱스를 생성하는 것이다. 뱅크 인덱스와 각 뱅크의 번지 생성은 FFT 길이가 2^n일 경우 n 비트 카운터를 이용하여 생성한다. 종래기술에서 설명된 Radix-4 알고리즘의 경우 2^2, 2^4, 2^6, 2^8 등의 포인트 수를 가진다. 그러므로 뱅크 인덱스(뱅크i) 생성은 2비트 디지트들에 대하여 모듈로-4 덧셈을 통해 구할 수 있다. 그러나, Mixed-radix의 경우 2^3, 2^5, 2^7, 2^9 등과 같은 포인트 수를 가져 입력 카운트 비트의 개수가 3비트, 5비트, 7비트, 9비트 등과 같이 홀수가 존재하게 된다. 따라서 2비트 디지트들의 모듈로-4 덧셈만으로는 뱅크 인덱스를 생성할 수 없다. 본 발명에 따른

Mixed-radix의 경우 입력 카운트 비트의 자리 수가 홀수인 경우 뱅크 인덱스를 생성하는 방법은 하기의 2가지 과정을 통해 수행된다. 첫째로, 최상위 2비트의 위치를 교환한다. 둘째로, 상기 위치 교환이 이루어진 카운트 값을 하위 비트부터 2비트 디지트들로 나누고 나뉘어진 2비트들에 대하여 모듈러-4 덧셈을 수행한다. 그러면 최상위 1비트가 남게 되는데, 남은 상기 최상위 1비트와 모듈러-4 덧셈에 의해 연산된 값을 모듈러-4로 다시 연산한다.

<60> 32(2^5)-포인트 FFT의 2비트 디지트들과 1비트를 구성하는 방법을 도 13에 도시하였다. 상술한 방법에 따라 입력 카운트 비트들의 위치를 교환하고 연산을 수행하게 되면, 모듈러-4의 연산을 수행할 때, 마지막으로 1비트가 남는 비트는 입력 데이터 카운트 비트 중 3번 비트에 해당한다. 그리고, 하위 비트부터 2비트 디지트씩 모듈러-4 덧셈 연산을 수행하는 비트들의 매칭은 (4, 2), (1, 0)에 해당한다. 이와 같이 뱅크를 생성하면 FFT 연산시 모두 다른 뱅크로부터 데이터를 가져올 수 있을 뿐만 아니라 도 5의 열2에서 열1로 변경 시 뱅크 인덱스 순서는 그대로 유지할 수 있다.

<61> 도 14는 32-포인트에서의 뱅크 인덱스 생성 방법을 일반화하여 2n-포인트 Mixed-radix FFT 일 때의 뱅크 인덱스 생성 방법을 나타낸 것이다. 도 14에서 입력 데이터 카운트 비트들 중 별도의 1비트 위치는 n-2번째 비트이며 2비트 디지트들의 모듈로-4 덧셈에 이 별도의 비트가 포함된다.

<62> 상술한 바와 같이 본 발명은 Radix-4 알고리즘에 기초한 Mixed-radix 알고리즘을 사용함으로써, 고속 연산이 가능해진다. 그리고, Mixed-radix 알고리즘에 인-플레이스 연산을 적용하고 입출력 동시 수행을 통해 4개의 뱅크로 구성된 N 워드 메모리 2개로 연속 처리를 수행함으로 인해 메모리에 사용되는 면적을 최소화 할 수 있다.

<63> 본 발명에 따른 FFT 프로세서의 연산 싸이클과 종래기술에서 설명된 FFT 프로세서의 연산 싸이클을 하기 <표 3>에 비교하여 도시하였다. 하기 <표 3>으로부터 Radix-2 알고리즘을 사용한 종래의 메모리 구조에 비해 연산 싸이클이 약 1/4로 감소됨을 알 수 있다.

<64> 【표 3】

구 조	알고리즘 클럭 싸이클	N = 2,048	N = 4,096
종래의 메모리 구조	Radix-2 $\frac{N}{2}$ log2N+2	11,266	24,578
본 발명의 구조	Radix-4 $\frac{N}{4}$ log4N+6	-	6,150
	Mixed-radix (Radix-4, Radix-2) $\frac{N}{4}$ log42N+6	3,078	-

<65> 또한, 다중 뱅크 구조를 사용하지 않은 Radix-4 알고리즘을 기반으로 한 종래의 Mixed-radix FFT 프로세서와 본 발명을 비교할 경우 종래기술에 따른 FFT 프로세서는 본 발명에 비해 약 4배 가까운 연산 싸이클이 소모된다.

<66> 뿐만 아니라 인-플레이스 알고리즘과 동시에 입출력하는 구조를 채택하지 않은 종래기술에서 설명된 Mixed-radix FFT 프로세서는 4개의 뱅크로 구성된 N 워드 메모리 4개를 요구하여 N 워드 메모리 2개를 요구하는 본 발명의 구조에 비해 2배의 메모리 면적을 가진다.

<67> 따라서, 본 발명은 고속 연산뿐만 아니라 적은 하드웨어 크기도 만족하여 OFDM, DMT 시스템에 적용하기 용이하다.

【발명의 효과】

<68> 상술한 바와 같이 본 발명을 이용하면, FFT 연산에서 고속 연산이 용이하며, 작은 크기의 하드웨어를 통해 구현할 수 있는 이점이 있다.

【특허청구범위】

【청구항 1】

입출력과 FFT 연산을 위한 메모리 중 입출력을 위한 메모리를 선택하고 메모리의 4개의 뱅크 중 하나의 뱅크를 선택하여 입력과 출력을 수행하기 위한 입출력 인터페이스와,

상기 인터페이스로의 입출력과 FFT 연산이 사용되는 4개의 뱅크로 구성된 2개의 N 워드 메모리와,

상기 인터페이스로의 입출력과 FFT 연산을 위한 메모리 중 FFT 연산을 위한 메모리를 선택하고 인-플레이스 연산을 위해 각 버터플라이 입출력에 할당된 뱅크들을 버터플라이 연산 회로의 4개의 입력과 연결하기 위한 제1데이터 교환부와,

상기 제1데이터 교환부에서 공급되는 Radix-4와 Radix-2 두 종류의 버터플라이를 하나의 회로로 수행 가능하며 대칭적인 리버스 출력을 구성하기 위한 버터플라이와,

상기 인터페이스부로의 입출력과 FFT 연산을 위한 메모리 중 FFT 연산을 위한 메모리를 선택하고 인-플레이스 연산을 위해 각 버터플라이 입출력에 할당된 뱅크들을 버터플라이 연산 회로의 4개의 출력과 연결하기 위한 제2데이터 교환부와,

상기 다중 뱅크 메모리 구조에서 인-플레이스 연산을 수행하기 위한 뱅크 인덱스 및 주소를 생성하기 위한 주소 생성부를 포함하는 것을 특징으로 하는 고속 퓨리에 변환을 이용한 혼합-기수 방식의 변조 장치.

【청구항 2】

제 1 항에 있어서,

상기 버터플라이는 Radix-4 버터플라이와 2개의 Radix-2 버터플라이의 출력을 교환하는 것에 의해 대칭적 리버스 출력 순서를 구성하여 순차적 입출력을 동시에 수행 가능하게 함으로써 4개의 뱅크로 구성된 N 워드 메모리 2개만으로 연속 처리를 수행할 수 있는 Radix-4/2 버터플라이 연산회로인 것을 특징으로 하는 고속 퓨리에 변환을 이용한 혼합-기수 방식의 변조 장치.

【청구항 3】

제 1 항에 있어서,

상기 인-플레이스 알고리즘은 Radix-4 알고리즘의 다중 뱅크 메모리 구조의인-플레이스 알고리즘을 변형하여 혼합-기수 알고리즘을 위한 인-플레이스 알고리즘인 것을 특징으로 하는 고속 퓨리에 변환을 이용한 혼합-기수 방식의 변조 장치.

【청구항 4】

제 1 항에 있어서,

상기 인-플레이스 알고리즘은 뱅크 인덱스 생성에서 2^n-포인트 연산일 때 n-2번째 비트를 별도로 모듈로-4 덧셈에 포함하는 것을 특징으로 하는 고속 퓨리에 변환을 이용한 혼합-기수 방식의 변조 장치.

【청구항 5】

고속 퓨리에 변환을 이용한 혼합-기수 방식의 변조 장치에 있어서,

입력 심볼을 저장하거나 고속 퓨리에 연산이 완료된 심볼들을 저장하는 4개의 뱅크로 구성된 2개의 메모리들과,

상기 메모리로부터 출력되는 심볼들의 수에 따라 기수-4 또는 기수-2 방식으로 버터플라이 연산을 수행하고, 상기 연산된 값들을 대칭적 리버스로 출력하는 버터플라이와,

상기 메모리들 중 하나의 메모리의 각 뱅크들로부터 각각 하나씩의 심볼들을 독취하여 상기 버터플라이로 출력하는 제1데이터 교환부와,

상기 버터플라이로부터 연산되어 출력된 심볼들을 상기 제1데이터 교환부에서 읽어온 주소와 동일한 주소에 저장되도록 각 심볼들을 매칭하는 제2데이터 교환부와,

상기 제1데이터 교환부에서 읽어온 심볼이 연산 이후 상기 제2데이터 교환부를 통해 출력될 시 상기 제1데이터 교환부에서 읽어온 뱅크 및 주소와 상기 제2데이터 교환부의 출력 뱅크 및 주소가 동일하도록 상기 제2데이터 교환부의 출력을 제어하는 주소 생성부를 포함함을 특징으로 하는 고속 퓨리에 변환을 이용한 혼합-기수 방식의 변조 장치.

【청구항 6】

제5항에 있어서,

상기 2개의 메모리와 입출력 데이터의 인터페이스를 수행하는 입출력 인터페이스를 더 포함함을 특징으로 하는 고속 퓨리에 변환을 이용한 혼합-기수 방식의 변조 장치.

【청구항 7】

제5항에 있어서, 상기 버터플라이는,

기수-4 버터플라이의 구조를 가지며, 기수-2로 동작이 필요한 경우 상기 버터플라이의 내부에 구비된 다중화기를 통해 2개의 기수-2 연산이 이루어지도록 구성됨을 특징으로 하는 고속 퓨리에 변환을 이용한 혼합-기수 방식의 변조 장치.

【청구항 8】

제5항에 있어서, 상기 버터플라이의 대칭적 리버스는,

상기 버터플라이 연산이 수행되는 심볼들의 총 개수에 결정되는 2진수의 출력 카운터 값을 2비트씩 하나의 단위로 하여 대칭적으로 변환하여 주소로 결정함을 특징으로 하는 고속 퓨리에 변환을 이용한 혼합-기수 방식의 변조 장치.

【청구항 9】

제8항에 있어서,

상기 2진수의 출력 카운터의 자리 수가 홀수인 경우 상기 자리 수의 중앙의 비트를 기준으로 대칭적으로 변환하여 주소로 결정함을 특징으로 하는 고속 퓨리에 변환을 이용한 혼합-기수 방식의 변조 장치.

【청구항 10】

제5항에 있어서, 상기 주소 생성부는,

상기 뱅크 결정 시, 상기 연산되는 심볼에 대응하는 2진수의 입력 카운트 비트 값을 모듈러-4 연산한 값으로 뱅크를 결정함을 특징으로 하는 고속 퓨리에 변환을 이용한 혼합-기수 방식의 변조 장치.

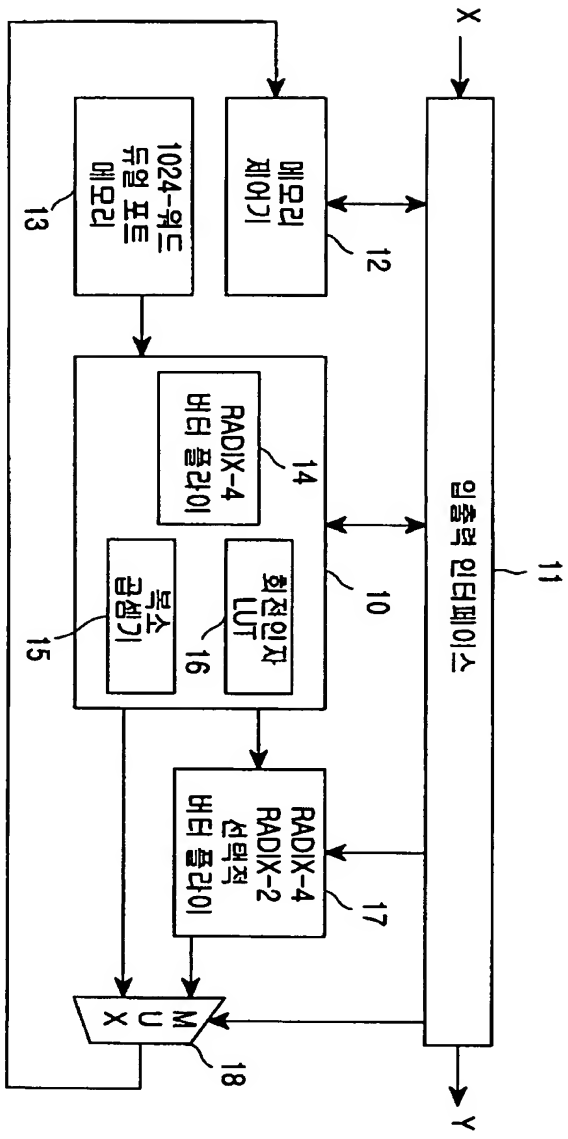
【청구항 11】

제10항에 있어서, 상기 주소 생성부는,

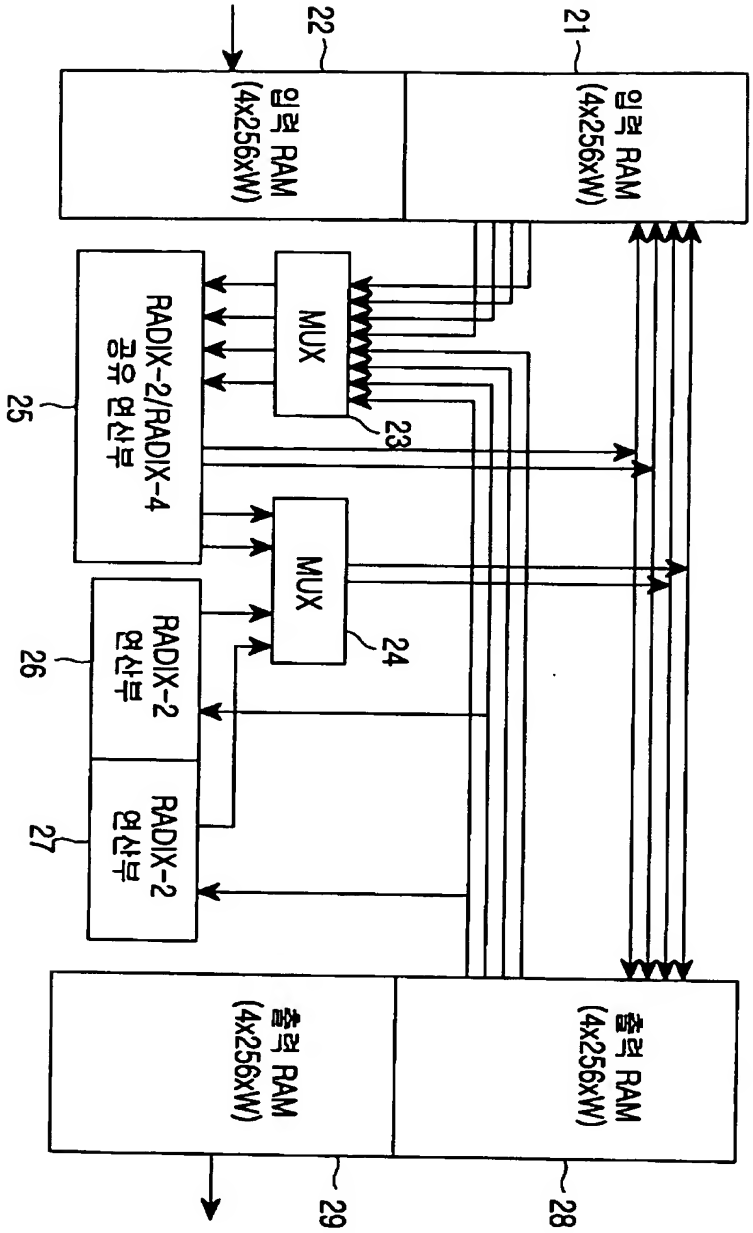
상기 2진수의 입력 카운트 비트의 자리 수가 홀수인 경우 최상위 2비트의 위치를 교환하고, 하위 2비트씩 모듈러-4연산을 수행한 후 교환된 최상위 비트와 모듈러-4 연산을 통해 뱅크를 결정함을 특징으로 하는 고속 퓨리에 변환을 이용한 혼합-기수 방식의 변조 장치.

【도면】

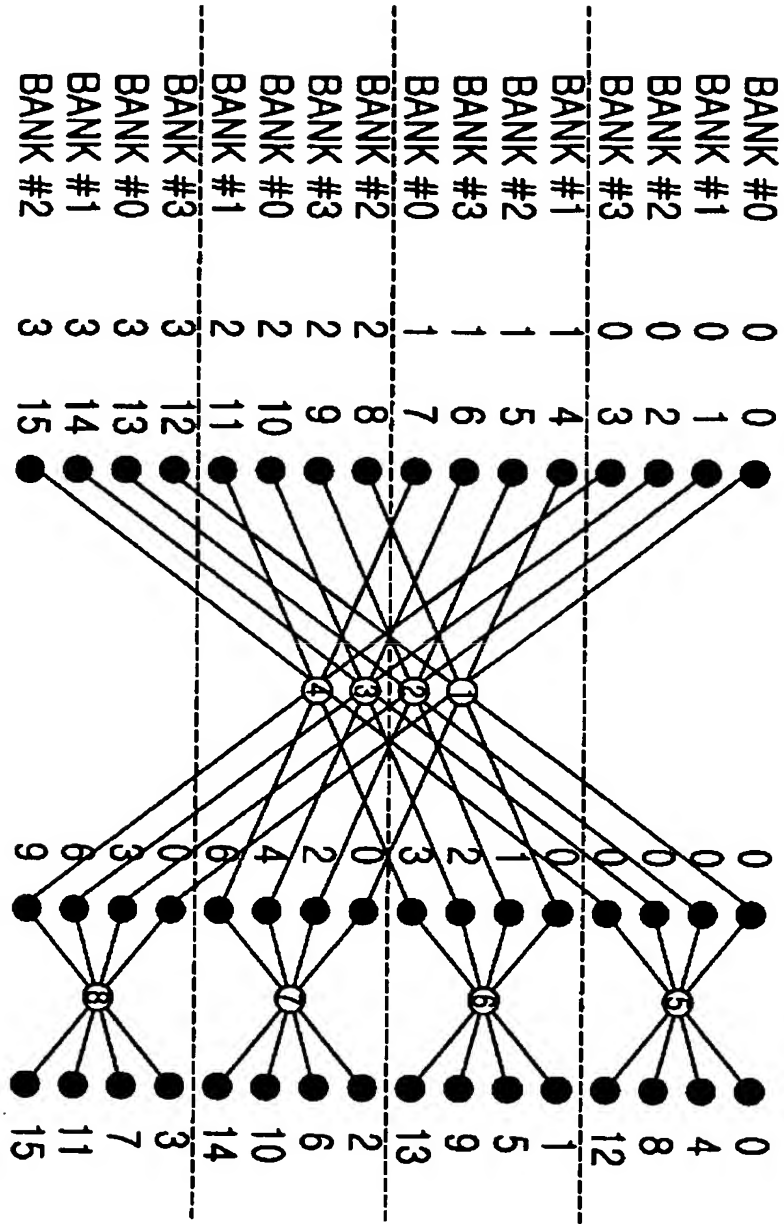
【도 1】


X
1024-워드 듀얼 포트 메모리
13
메모리 제어기
12
RADIX-4 버터플라이
14
복소 곱셈기
15
회전인자 LUT
16
10
입출력 인터페이스
11
RADIX-4 RADIX-2 선택적 버터플라이
17
MUX
18
Y

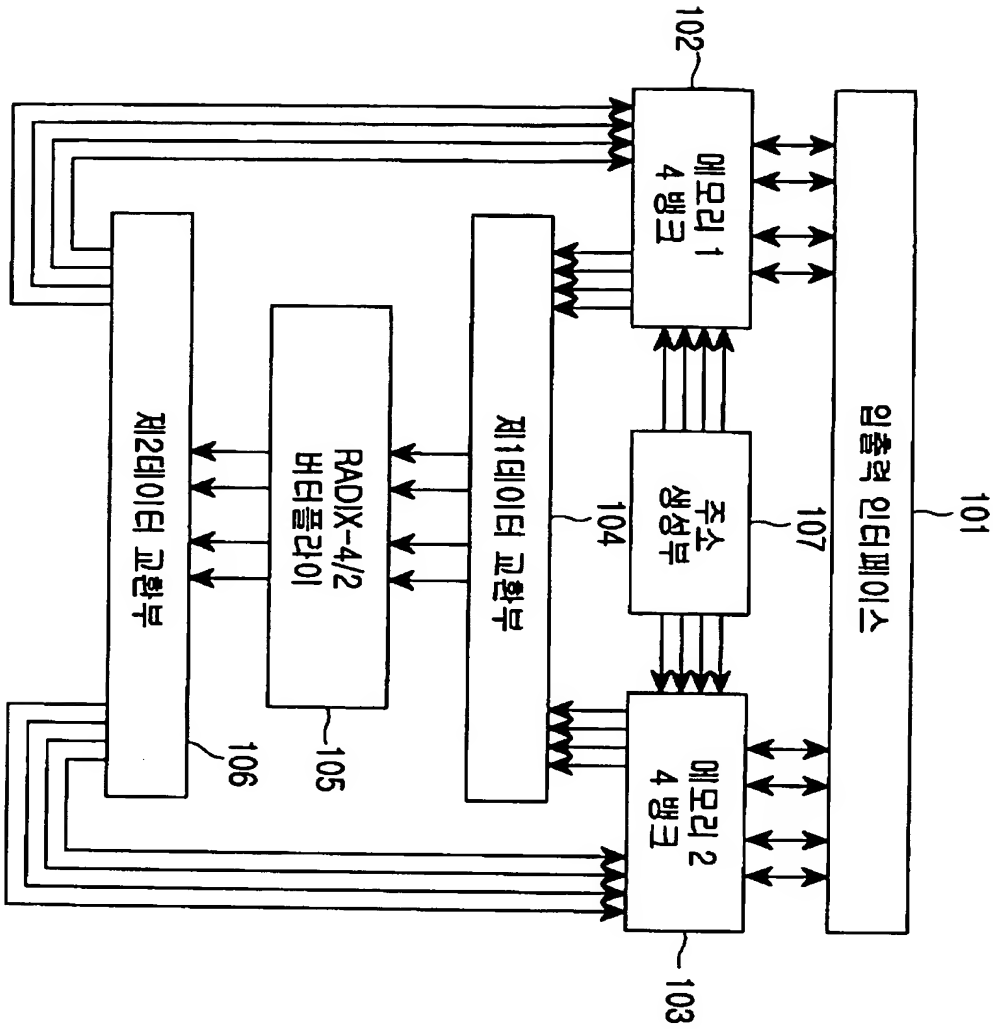
【도 2】


22
21
입력 RAM
(4x256xW)
입력 RAM
(4x256xW)
RADIX-2/RADIX-4
공유 연산부
25
MUX
23
MUX
24
RADIX-2
연산부
26
RADIX-2
연산부
27
출력 RAM
(4x256xW)
출력 RAM
(4x256xW)
29
28

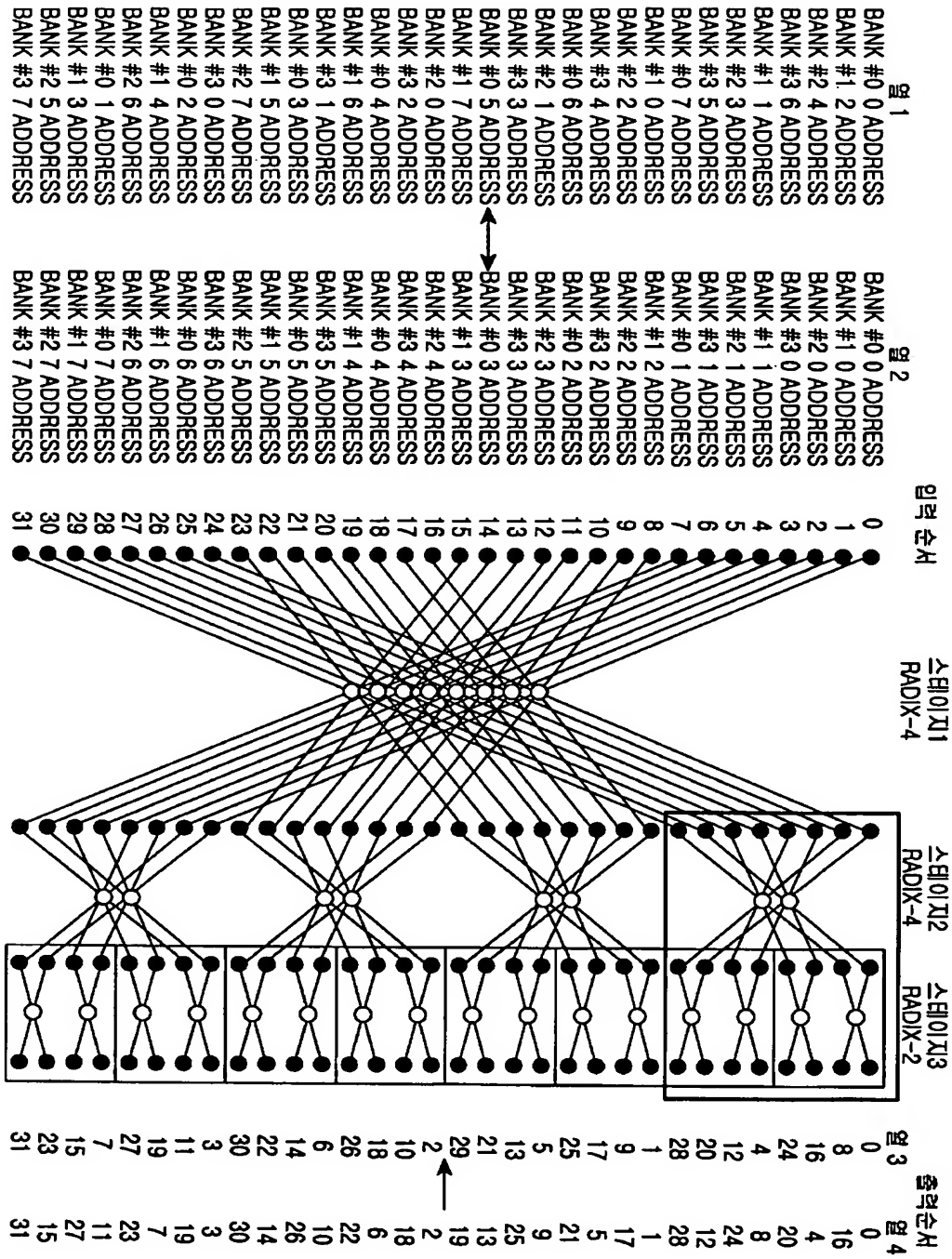
【도 3】


BANK #0
BANK #1
BANK #2
BANK #3
BANK #1
BANK #2
BANK #3
BANK #0
BANK #2
BANK #3
BANK #0
BANK #1
BANK #3
BANK #0
BANK #1
BANK #2
0 0 0 0 1 1 1 1 2 2 2 2 3 3 3 3
0 2 3 4 5 6 7 8 9 10 11 12 13 14 15
1 2 3 4
0 0 0 0 1 2 3 3 0 2 4 6 0 3 6 9
5 6 7 8
0 4 8 12 1 5 9 13 2 6 10 14 3 7 11 15

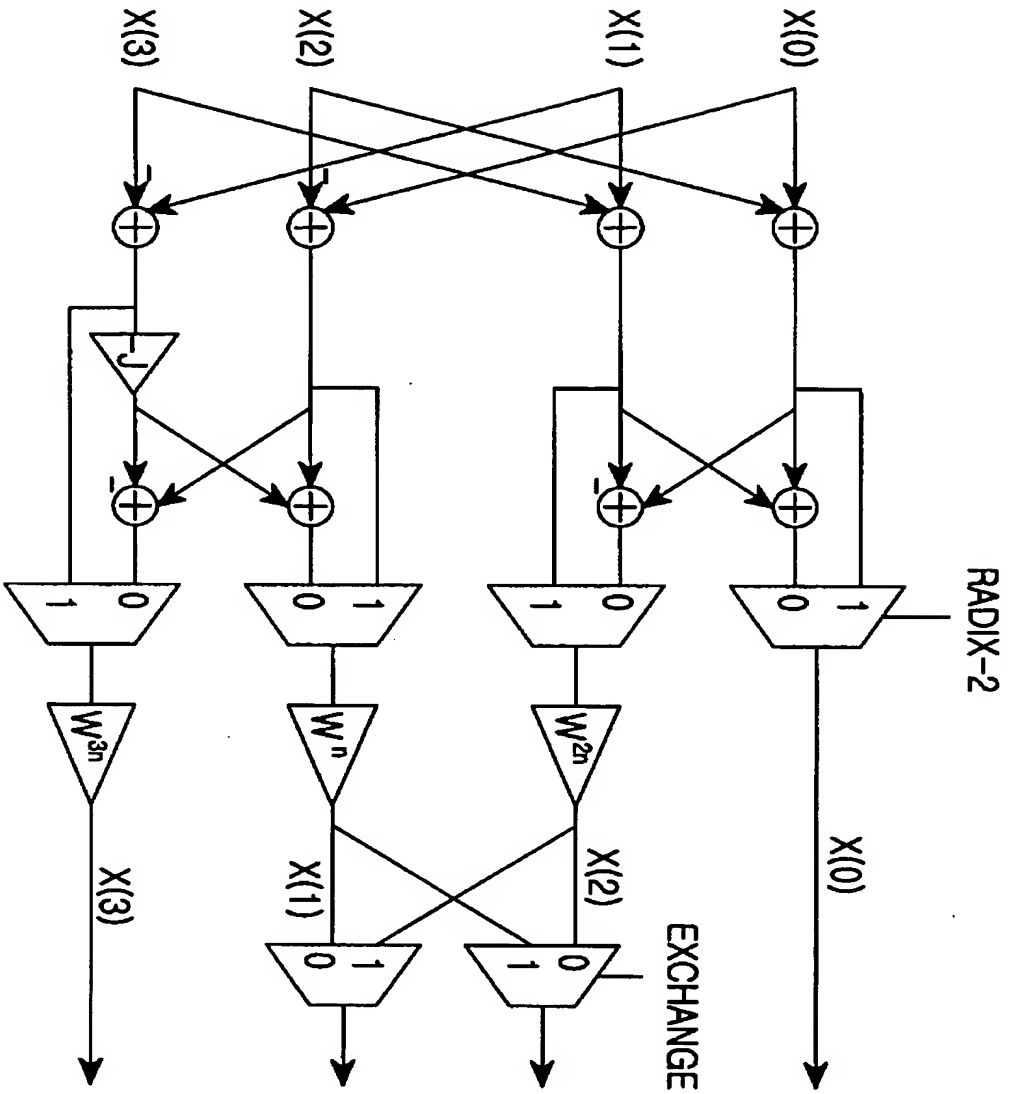
【도 4】


102
메모리 1
4 뱅크
입출력 인터페이스
101
제2데이터 교환부
RADIX-4/2
버터플라이
제1데이터 교환부
104
주소
생성부
107
106
105
메모리 2
4 뱅크
103

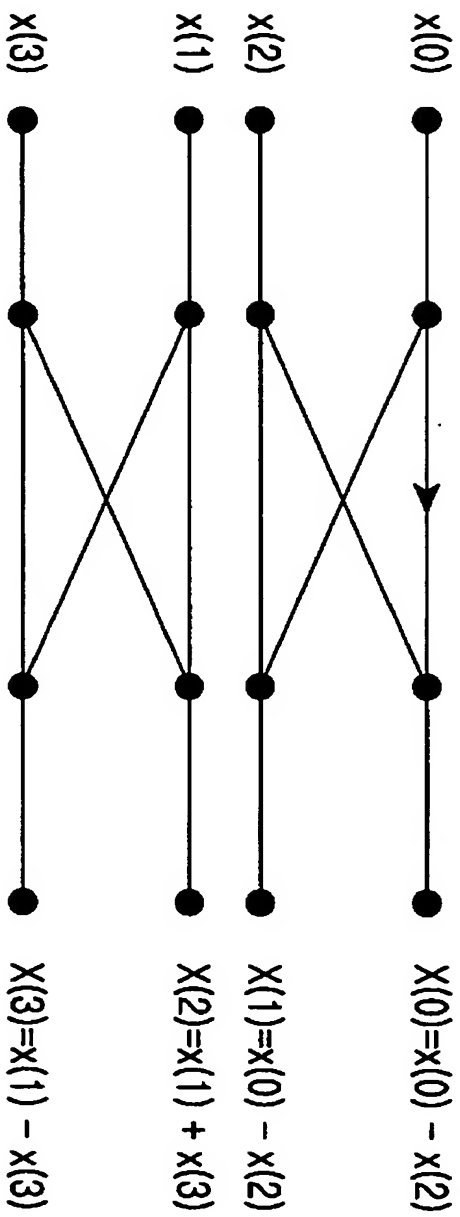
【도 5】


열 1
BANK #0 0 ADDRESS
BANK #1 2 ADDRESS
BANK #2 4 ADDRESS
BANK #3 6 ADDRESS
BANK #1 1 ADDRESS
BANK #2 3 ADDRESS
BANK #3 5 ADDRESS
BANK #0 7 ADDRESS
BANK #1 0 ADDRESS
BANK #2 2 ADDRESS
BANK #3 4 ADDRESS
BANK #0 6 ADDRESS
BANK #2 1 ADDRESS
BANK #3 3 ADDRESS
BANK #0 5 ADDRESS
BANK #1 7 ADDRESS
BANK #2 0 ADDRESS
BANK #3 2 ADDRESS
BANK #0 4 ADDRESS
BANK #1 6 ADDRESS
BANK #3 1 ADDRESS
BANK #0 3 ADDRESS
BANK #1 5 ADDRESS
BANK #2 7 ADDRESS
BANK #3 0 ADDRESS
BANK #0 2 ADDRESS
BANK #1 4 ADDRESS
BANK #2 6 ADDRESS
BANK #0 1 ADDRESS
BANK #1 3 ADDRESS
BANK #2 5 ADDRESS
BANK #3 7 ADDRESS
열 2
입력 순서
스테이지1 RADIX-4
스테이지2 RADIX-4
스테이지3 RADIX-2
열 3
열 4
출력순서

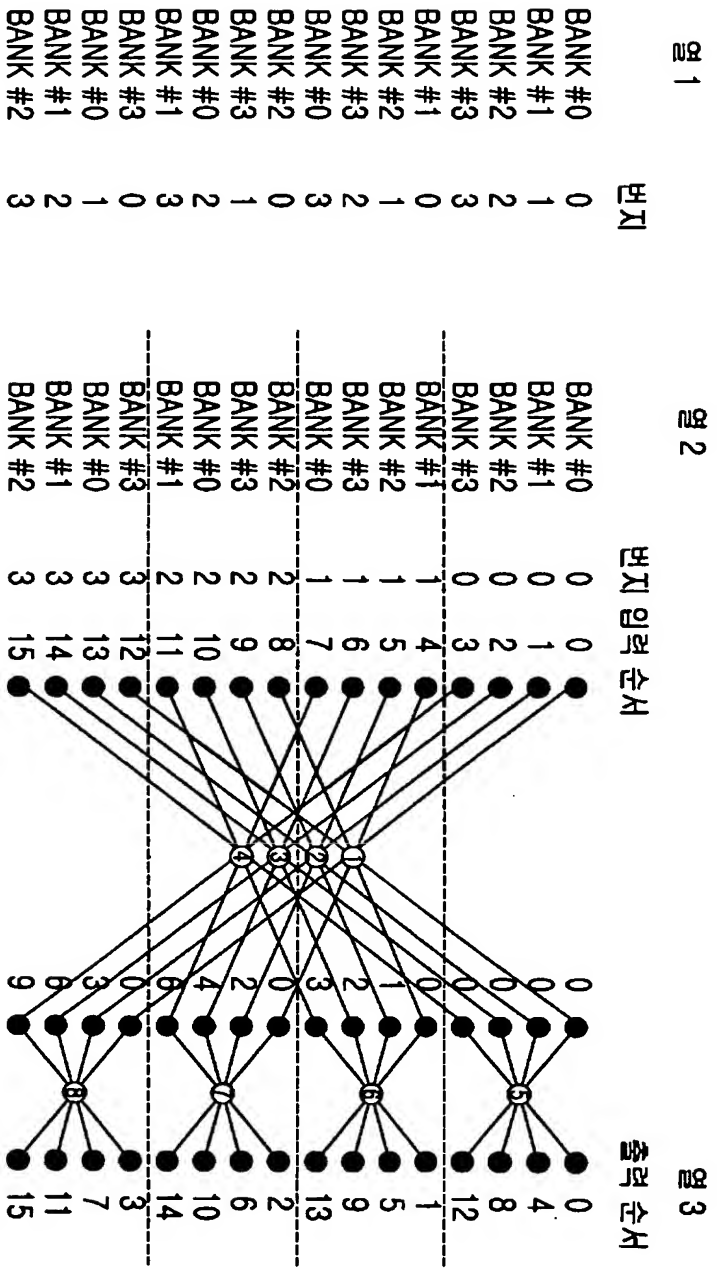
【도 6a】



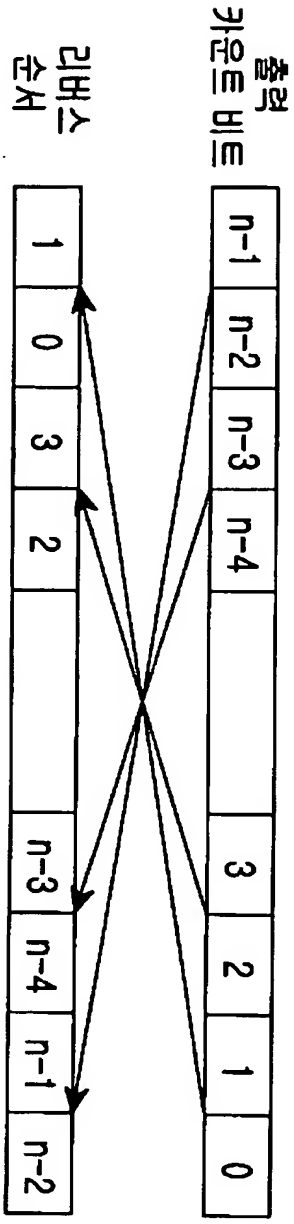
【도 6b】



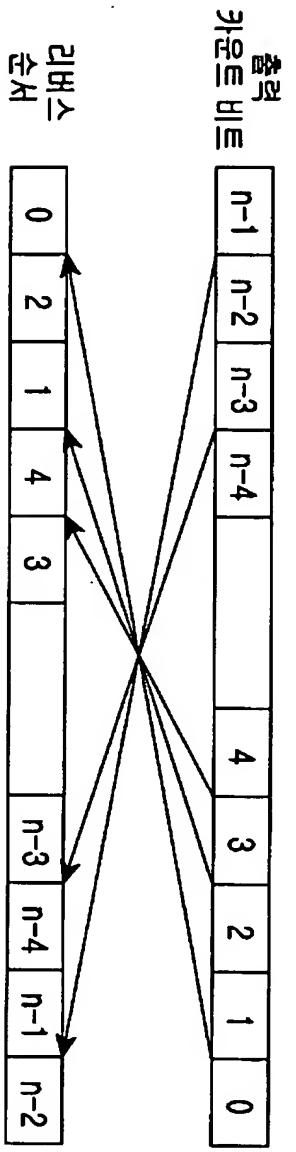
【도 7】


열 1
뱅지
BANK #0 0
BANK #1 1
BANK #2 2
BANK #3 3
BANK #1 0
BANK #2 1
BANK #3 2
BANK #0 3
BANK #2 0
BANK #3 1
BANK #0 2
BANK #1 3
BANK #3 0
BANK #0 1
BANK #1 2
BANK #2 3
열 2
뱅지 입력 순서
BANK #0 0 0
BANK #1 0 1
BANK #2 0 2
BANK #3 0 3
BANK #1 1 4
BANK #2 1 5
BANK #3 1 6
BANK #0 1 7
BANK #2 2 8
BANK #3 2 9
BANK #0 2 10
BANK #1 2 11
BANK #3 3 12
BANK #0 3 13
BANK #1 3 14
BANK #2 3 15
열 3
출력 순서
0 0
4 0
8 0
12 0
1 1
5 2
9 1
13 3
2 2
6 0
10 4
14 2
3 3
7 0
11 6
15 6
9

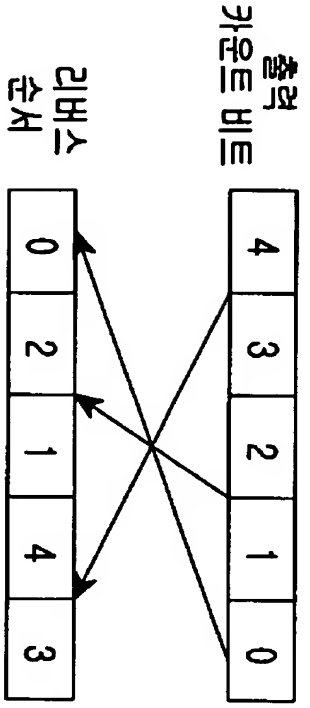
【도 8a】



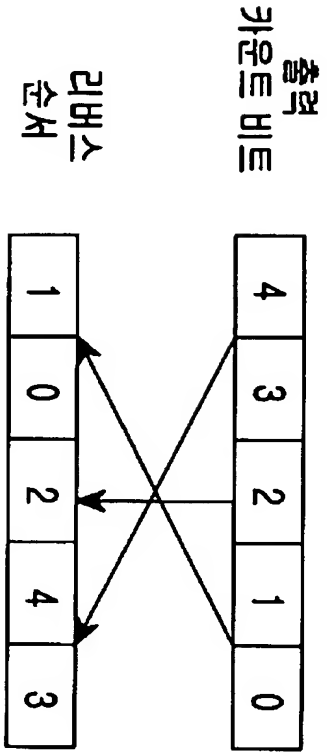
【도 8b】



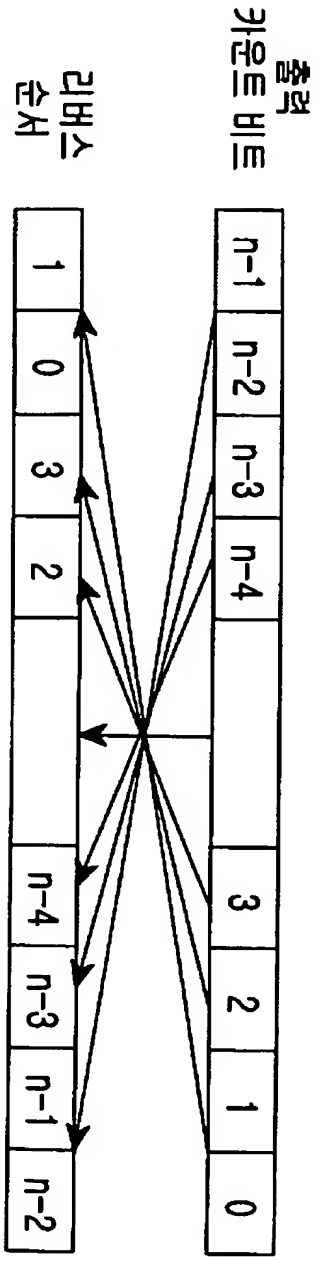
【도 9】



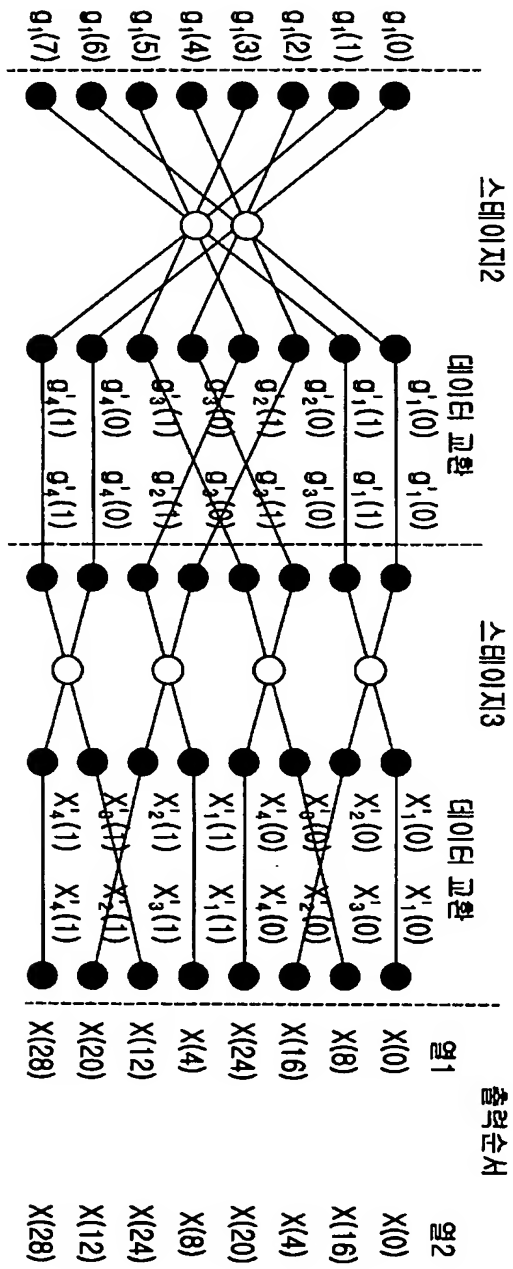
【도 10】



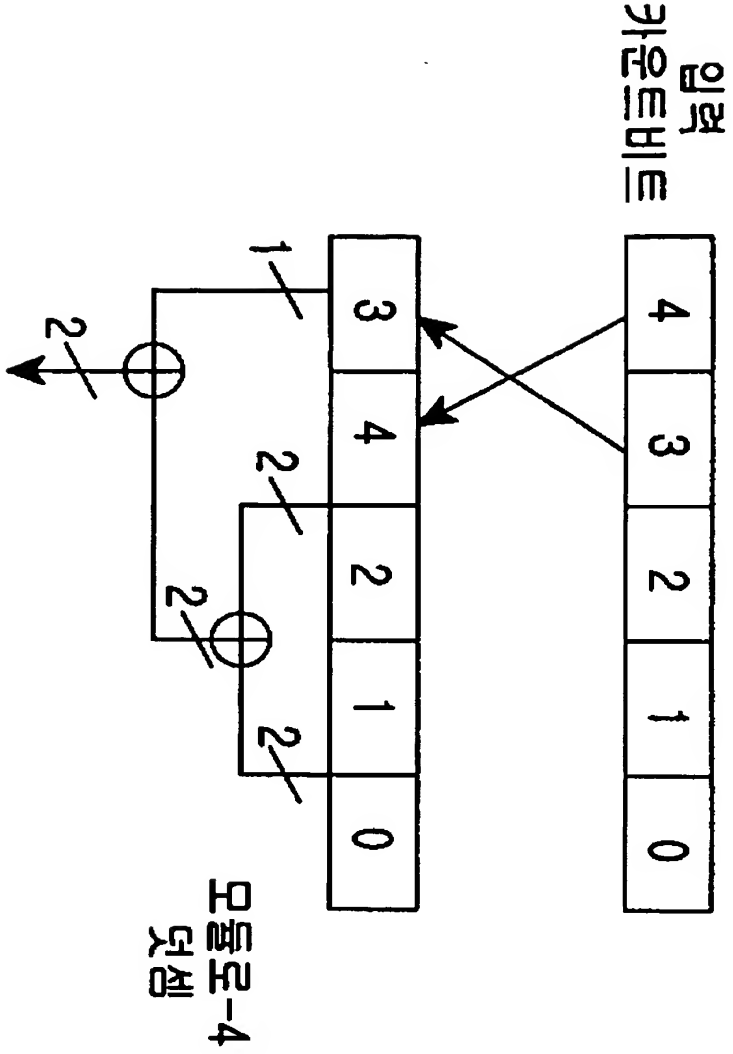
【도 11】



【도 12】


g1(0)
g1(1)
g1(2)
g1(3)
g1(4)
g1(5)
g1(6)
g1(7)
스테이지2
데이터 교환
g'1(0) g'1(1)
g'1(1) g'1(1)
g'2(0) g'3(0)
g'3(1) g'2(1)
g'3(0) g'3(0)
g'2(1) g'4(1)
g'4(0) g'4(0)
g'4(1) g'4(1)
스테이지3
데이터 교환
X'1(0) X'1(0)
X'2(0) X'3(0)
X'3(0) X'2(0)
X'4(0) X'4(0)
X'1(1) X'1(1)
X'2(1) X'3(1)
X'3(1) X'2(1)
X'4(1) X'4(1)
출력순서
열1
X(0)
X(8)
X(16)
X(4)
X(24)
X(20)
X(12)
X(28)
열2
X(0)
X(16)
X(8)
X(24)
X(4)
X(20)
X(12)
X(28)

【도 13】



【도 14】